МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Факультет програмної інженерії та бізнесу

Кафедра інженерії програмного забезпечення

Практичні роботи

Minor *«Розробник ігрових додатків»*

дисципліна *«Програмування віртуальної реальності»*

(назва дисципліни)

Виконав: студент 4 курсу групи 632ПСТ

напряму підготовки (спеціальності):

121 інженерія програмного забезпечення

(шифр і назва напряму підготовки / спеціальності)

Бреславець М.О.

(прізвище й ініціали студента)

Прийняв: *доц. каф 603, к.т.н, Лучшев П.О.*

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала:

Кількість балів:

Оцінка ECTS:

Зміст

[1. Формування простору віртуальної реальності 3](#_Toc184219843)

[1.1. Моделювання та генерація ландшафту 3](#_Toc184219844)

[1.2. Моделювання поверхні води 7](#_Toc184219845)

[1.3. Моделювання атмосфери, туману 14](#_Toc184219846)

[2. Фотореалістичність об'єктів віртуальної реальності 19](#_Toc184219847)

[2.1. Перспективна проекція та управління точкою зору 19](#_Toc184219848)

[2.2. Використання текстур 19](#_Toc184219849)

[3. Створення та моделювання об'єктів віртуальної реальності 21](#_Toc184219850)

[3.1. Моделювання об’єктів віртуальної реальності. 21](#_Toc184219851)

[4. Сприйняття віртуальної реальності людиною 26](#_Toc184219852)

[4.1. Наближення створеного продукту до VR 26](#_Toc184219853)

[5. Зв'язок віртуального та реального світу 30](#_Toc184219854)

[5.1. Назва роботи 30](#_Toc184219855)

[Загальний перелік посилань 31](#_Toc184219856)

[Додаток А. Лістинг програми до практичної роботи № 1.1 32](#_Toc184219857)

[Додаток Б. Лістинг програми до практичної роботи № 2.1 37](#_Toc184219858)

[Додаток ?. Лістинг програми до практичної роботи №? 39](#_Toc184219859)

# 1. Формування простору віртуальної реальності

## 1.1. Моделювання та генерація ландшафту

порп

## 1.2. Моделювання поверхні води

прпрр

## 1.3. Моделювання атмосфери, туману

прпр

# 2. Фотореалістичність об'єктів віртуальної реальності

## 2.1. Перспективна проекція та управління точкою зору

апрпрр

## 2.2. Використання текстур

роророр

# 3. Створення та моделювання об'єктів віртуальної реальності

## 3.1. Моделювання об’єктів віртуальної реальності.

роророро

# 4. Сприйняття віртуальної реальності людиною

## 4.1. Наближення створеного продукту до VR

### Завдання

Зробити можливість, показу програмного додатку, начебто його запустили і використовують в середовищі VR.

### Результати виконання практичної роботи

### Для виконання поставленого завдання потрібно зробити так, щоб камера в додатку була не з третього режиму (режим коли видно і користувача і світ який він бачить), а із першого режиму (режим з першого лиця – нібито від очей).

Взагалі для зміни налаштування камери, тобто її відображення, так як вона вже має достатньо потужний компонент «Cinemaсhine», потрібно просто налаштувати його, після налаштувань ми отримали такий результат відображення ігрового середовища (камера знаходиться ніби біля очей гравця), це і продемонстровано на рисунок 32.

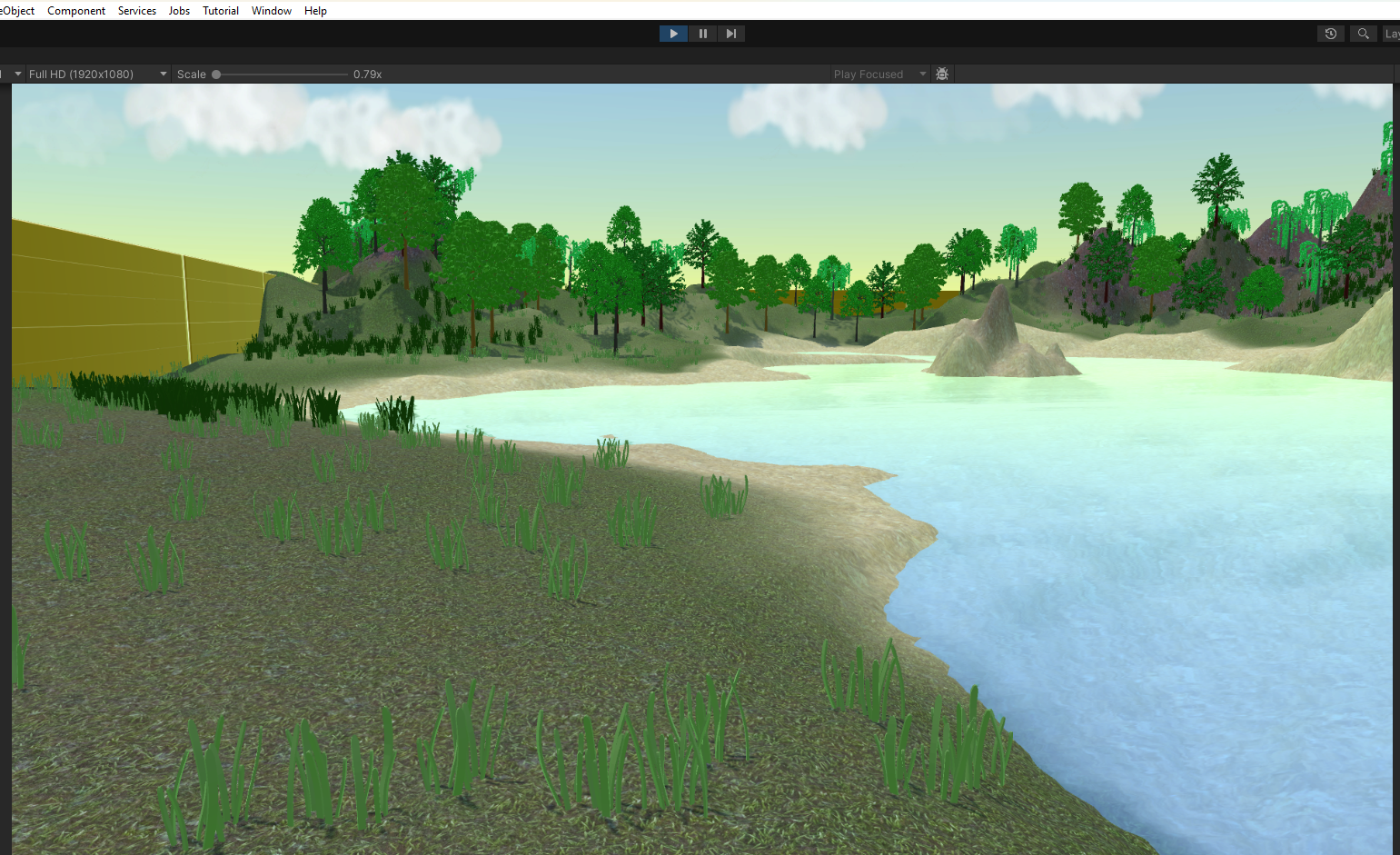


Рисунок 32 – Новий режим відображення ігрового світу.

### Тепер гра стала ще біль наближена до реального світу, але мені здалося це замалим, я захотів втілити таку ідею, щоб гравець як і раніше ходив з третього режиму (видно і грався і світ), але на екрані ще було віконечко яке відповідало начебто як за зображення отримане із правого та лівого ока.

Давно я вже хотів втілити в свої проекти щось подібне, але так завжди і не наважувався, просто заміняв, тому що не знав як це реалізувати ?

Про що взагалі йде мова, вона іде за те що, на ігровому полі є ще одна, або декілька камер, але вони відображається в певній частині екрану, або на певному об’єкті, наприклад, мапа гри, камера заднього вигляду авто, яка відображається коли користувач ввімкнув задню передачу, або телевізор, який відображає інші камери відео спостережень, розміщені в будь якому місці ландшафту гри.

І ось я знайшов як це реалізувати. Почнемо, спочатку додаємо камеру на сцену, потім налаштовуємо її, робимо те зображення , яке ми хочемо побачити.

Далі для її відображення, в ієрархії файлів, створюємо елемент «[*Render Texture*](http://docs.unity3d.com/Manual/class-RenderTexture.html)».

В налаштування камери вказуємо в поле «*Target Texture*», створений на попередньому кроці елемент, результат цього кроку наведений на рисунку 33.

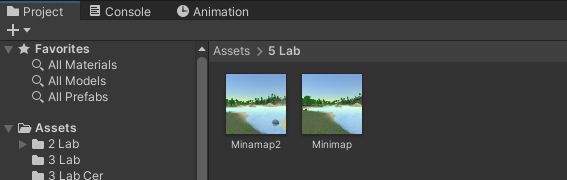


Рисунок 33 - Створені дві «[*Render Texture*](http://docs.unity3d.com/Manual/class-RenderTexture.html)*»*

Та для відображення вигляду камер користувачу, в графічний об’єкт «Canvas», добавляємо навий об’єкт UI «Plane», Далі додаємо ще 2 елемента «Image», але сам компонент «Image», замінюймо на «Raw Image», та в рядок «Texture», переміщуємо елемент «[*Render Texture*](http://docs.unity3d.com/Manual/class-RenderTexture.html)». Створеного та налаштованого компоненту рисунок 34.

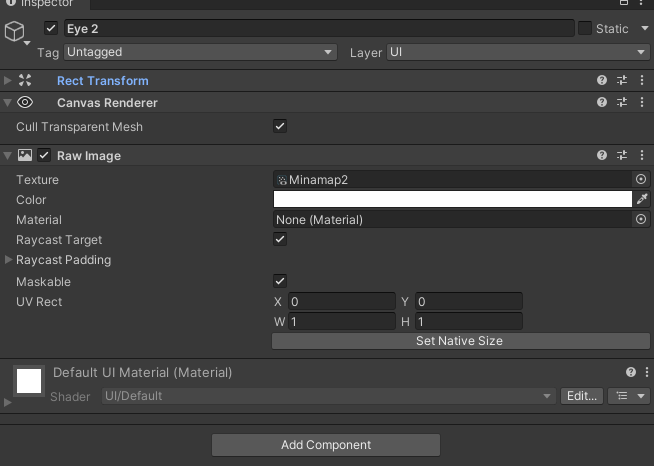


Рисунок 34 – Налаштований компонент відображення

На рисунку 35-36 – Зображений вигляд ігрового сеансу в режимі гри.

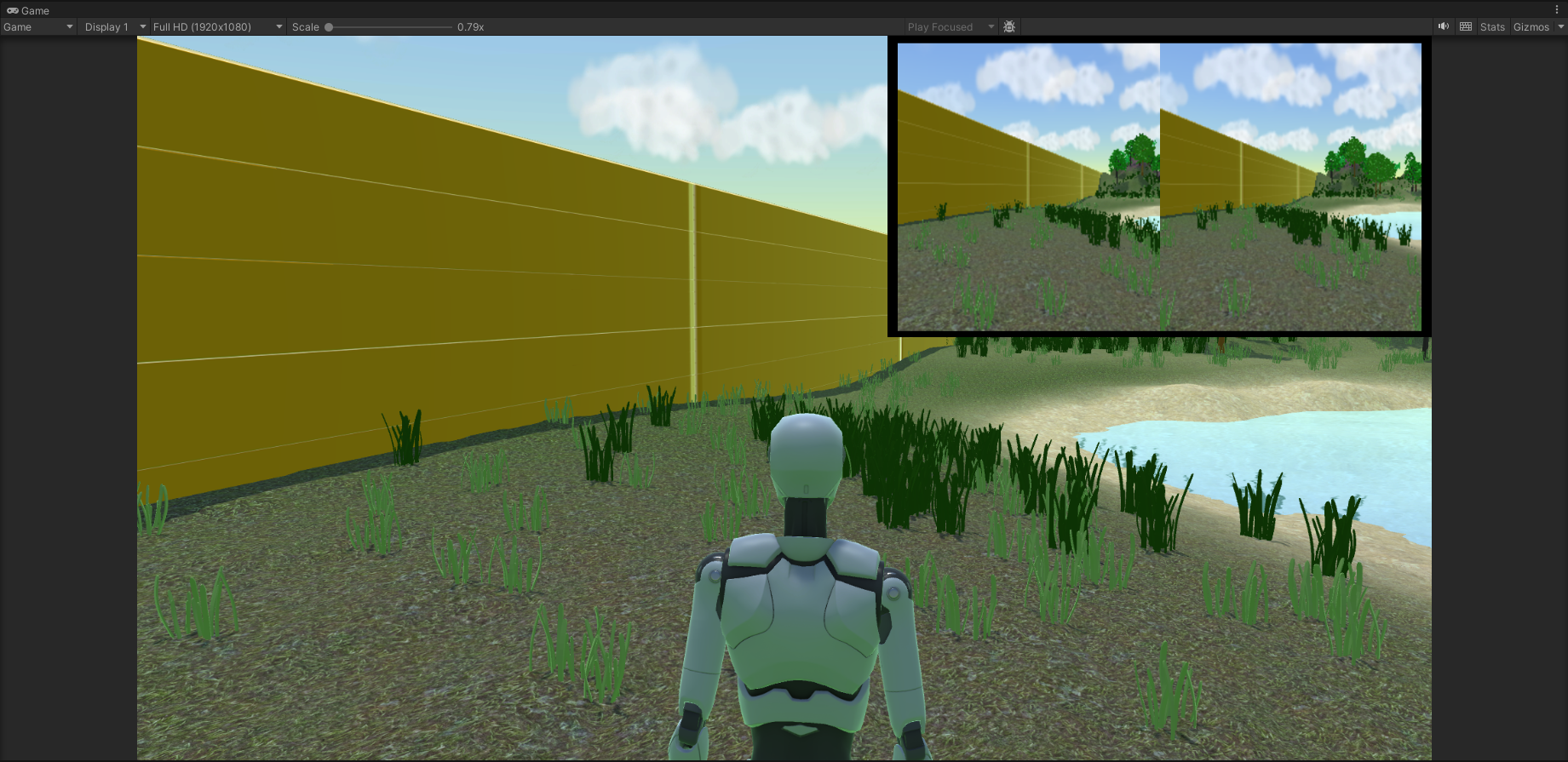


Рисунок 35 – Вигляд зображення, яке бачить користувач



Рисунок 36 – Вигляд зображення, яке бачить користувач

### Контроль виконання вимог та елементів завдання

В результаті виконання практичної роботи були повністю виконані елементи базового рівня та частково підвищеного рівня складності, що відображено в таблиці 4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблиця 4 | | | | |
| № з/п | Складність | Вимоги | Бали | Зроблено |
|  | Базовий рівень | Налаштування камери ,вигляд від 1 режиму (від очей гравця) | 2 | **2** |
|  | Реалізований рух камери разом із гравцем. | 2 | **2** |
|  | Підвищений рівень | Використання ігрових компоненту «[Render Texture](http://docs.unity3d.com/Manual/class-RenderTexture.html)» | 2 | **2** |
|  | Реалізація 2 камер, які відображаються в режимі гри (зображення ніби від очей) | 4 | **4** |

# 5. Зв'язок віртуального та реального світу

## 5.1. Назва роботи

### Завдання

### Теоретичні відомості

### Результати виконання практичної роботи

### Контроль виконання вимог та елементів завдання

# Загальний перелік посилань

1. Microsoft. glDrawArrays function [Електронний ресурс] / Microsoft – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/windows/win32/opengl/gldrawarrays>.
2. Diamond-square algorithm [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Diamond-square_algorithm>.

# Додаток А. Лістинг програми до практичної роботи № 1.1

### Код файлу (TerreinGen.cs)

using System.Collections;

using System.Collections.Generic;

using UnityEngine.UI;

using UnityEngine;

public class TerreinGen : MonoBehaviour

{

public Text widthCan, heightCan, scaleCan, offsetXCan, offsetYCan, noiseIntensityCan;

private int width = 512; // Width of the terrain

private int height = 512; // Height of the terrain

private float scale = 10f; // Scale of the terrain

private float offsetX = 100f; // X offset for noise

private float offsetY = 100f; // Y offset for noise

private float noiseIntensity = 0.1f; //Intensity of the noise

Terrain terrain;

//Renderer render;

// public Material material;

private void Start()

{

terrain = GetComponent<Terrain>();

//render = GetComponent<Renderer>();

//render.material = material;

NewTerrain();

OnRenameInfo();

}

private void NewTerrain() {

// Створення нового екземпляра TerrainData

TerrainData terrainData = new TerrainData();

// Встановити роздільну здатність карти висот і розмір TerrainData

terrainData.heightmapResolution = width;

terrainData.size = new Vector3(width, 600, height);

// Створення висот рельєфу

float[,] heights = GenerateHeights();

terrainData.SetHeights(0, 0, heights);

terrain.terrainData = terrainData;

}

private void OnRenameInfo()

{

widthCan.text = width.ToString();

heightCan.text = height.ToString();

scaleCan.text = scale.ToString();

offsetXCan.text = offsetX.ToString();

offsetYCan.text = offsetY.ToString();

noiseIntensityCan.text = noiseIntensity.ToString();

}

private float[,] GenerateHeights()

{

float[,] heights = new float[width, height];

for (int x = 0; x < width; x++)

{

for (int y = 0; y < height; y++)

{

// Створення значення шуму Perlin для поточної позиції

float xCoord = (float)x / width \* scale + offsetX;

float yCoord = (float)y / height \* scale + offsetY;

float noiseValue = Mathf.PerlinNoise(xCoord, yCoord);

//Встановіть висоту рельєфу на основі рівня шуму

heights[x, y] = noiseValue \* noiseIntensity;

}

}

return heights;

}

public void ButtonClick(bool triger, int inf)

{

switch (inf)

{

case 3: if (triger) scale++; else scale--; break;

case 4: if (triger) offsetX++; else offsetX--; break;

case 5: if (triger) offsetY++; else offsetY--; break;

case 6: if (triger) noiseIntensity += 0.01f; else noiseIntensity -= 0.01f; break;

}

NewTerrain();

OnRenameInfo();

}

public void ButtonNext(int inf)

{

ButtonClick(true, inf);

}

public void ButtonPrev(int inf)

{

ButtonClick(false, inf);

}

}

### Код файлу (MeshGenerator.cs)

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UIElements;

[RequireComponent(typeof(MeshFilter))]

public class MeshGenerator : MonoBehaviour

{

Mesh mesh;

Vector3[] vertices;

int[] triangles;

Color[] colors;

public int xSize = 20, zSize = 20;

public float NoiseIntensetive = 2f;

public Gradient gradient;

float minTerrainHeiht, maxTerrainHeiht;

void Start()

{

mesh = new Mesh();

GetComponent<MeshFilter>().mesh = mesh;

OnUpdateMesh();

}

public void OnUpdateMesh()

{

CreateShape();

UpdateMesh();

}

void CreateShape()

{

vertices = new Vector3[(xSize + 1) \* (zSize + 1)];

for(int i=0, z = 0; z <= zSize; z++)

{

for(int x=0;x <= xSize; x++)

{

float y = Mathf.PerlinNoise(x \* .1f, z \* .1f) \* NoiseIntensetive;

vertices[i] = new Vector3(x, y, z);

if(y>maxTerrainHeiht)

maxTerrainHeiht = y;

if(y < minTerrainHeiht)

minTerrainHeiht = y;

i++;

}

}

triangles = new int[xSize \* zSize \* 6];

int vert = 0;

int tris = 0;

for (int z = 0; z < zSize; z++)

{

for (int x = 0; x < xSize; x++)

{

triangles[tris + 0] = vert + 0;

triangles[tris + 1] = vert + xSize + 1;

triangles[tris + 2] = vert + 1;

triangles[tris + 3] = vert + 1;

triangles[tris + 4] = vert + xSize + 1;

triangles[tris + 5] = vert + xSize + 2;

vert++;

tris += 6;

}

vert++;

}

colors = new Color[vertices.Length];

for (int i = 0, z = 0; z <= zSize; z++)

{

for (int x = 0; x <= xSize; x++)

{

float heidht =Mathf.InverseLerp(minTerrainHeiht, maxTerrainHeiht, vertices[i].y);

colors[i] = gradient.Evaluate(heidht);

i++;

}

}

}

void UpdateMesh()

{

mesh.Clear();

mesh.vertices = vertices;

mesh.triangles = triangles;

mesh.colors = colors;

mesh.RecalculateNormals();

}

private void OnDrawGizmos()

{

if(vertices == null)

return;

for(int i=0;i<vertices.Length; i++)

{

Gizmos.DrawSphere(vertices[i], .1f);

}

}

}

### Код файлу (MeshControlPanel.cs)

using UnityEngine;

using UnityEngine.UI;

public class MeshControlPanel : MonoBehaviour

{

public Text textXSize, textZSize, textNoiseIntensetive;

MeshGenerator meshGen;

void Start()

{

meshGen = this.GetComponent<MeshGenerator>();

UpdateTextConvas();

}

void UpdateTextConvas()

{

textXSize.text = meshGen.xSize.ToString();

textZSize.text = meshGen.zSize.ToString();

textNoiseIntensetive.text = meshGen.NoiseIntensetive.ToString();

int max = 10;

if(meshGen.xSize > meshGen.zSize)

max = meshGen.xSize;

else

max = meshGen.zSize;

Camera.main.transform.position = new Vector3(Camera.main.transform.position.x, max, Camera.main.transform.position.z);

}

public void ButtonClickNext(int inf){

switch (inf) {

case 1: if(meshGen.xSize < 100) meshGen.xSize += 5; break;

case 2: if (meshGen.zSize < 100) meshGen.zSize += 5;break;

case 3: if (meshGen.NoiseIntensetive < 15) meshGen.NoiseIntensetive+=1f; break;

}

meshGen.OnUpdateMesh();

UpdateTextConvas();

}

public void ButtonClickPrev(int inf){

switch (inf)

{

case 1: if (meshGen.xSize > 10) meshGen.xSize -= 5; break;

case 2: if (meshGen.zSize > 10) meshGen.zSize -= 5; break;

case 3: if (meshGen.NoiseIntensetive > 2) meshGen.NoiseIntensetive -= 1f; break;

}

meshGen.OnUpdateMesh();

UpdateTextConvas();

}

}

# Додаток Б. Лістинг програми до практичної роботи № 2.1

### Код файлу (PanelControlWater.cs)

using UnityEngine.UI;

using UnityEngine;

using System;

public class PanelControlWater : MonoBehaviour

{

public float WaterLerp, NormalStrength, Refraction, Lerp;

private string waterLerpText, normalStrengthText, refractionText, lerpText;

public GameObject Water;

private Material material;

void Start()

{

//ініціалізація змінних

material = Water.GetComponent<Renderer>().material;

Debug.Log(material.GetFloat("\_Water\_Lerp").ToString());

waterLerpText = material.GetFloat("\_Water\_Lerp").ToString();

normalStrengthText = material.GetFloat("\_Normal\_Strength").ToString();

refractionText = material.GetFloat("\_Refraction").ToString();

lerpText = material.GetFloat("\_Lerp").ToString();

//Запис настоящих даних змінні

OnReadTextInSystem();

//Виведення змінних системи на екран

PrintTextOnScreen();

}

public void ButtonNext(int id){

switch (id){

case 0: if (WaterLerp < 1) WaterLerp += 0.05f; break;

case 1: if (NormalStrength < 2)NormalStrength += 0.05f; break;

case 2: if (Refraction < 2) Refraction += 0.05f; break;

case 3: if (Lerp < 1)Lerp += 0.05f; break;

}

OnRaitTextSheider();

PrintTextOnScreen();

}

public void ButtonPrevius(int id)

{

switch (id)

{

case 0: if (WaterLerp > 0) WaterLerp -= 0.05f; break;

case 1: if (NormalStrength > 0) NormalStrength -= 0.05f; break;

case 2: if (Refraction > 0) Refraction -= 0.05f; break;

case 3: if (Lerp > 0) Lerp -= 0.05f; break;

}

OnRaitTextSheider();

PrintTextOnScreen();

}

private void OnRaitTextSheider()

{

material.SetFloat("\_Water\_Lerp", WaterLerp);

material.SetFloat("\_Normal\_Strength", NormalStrength);

material.SetFloat("\_Refraction", Refraction);

material.SetFloat("\_Lerp", Lerp);

}

private void OnReadTextInSystem()

{

WaterLerp = material.GetFloat("\_Water\_Lerp");

NormalStrength = material.GetFloat("\_Normal\_Strength");

Refraction = material.GetFloat("\_Refraction");

Lerp = material.GetFloat("\_Lerp");

}

private void PrintTextOnScreen()

{

this.transform.GetChild(1).GetComponent<Text>().text = WaterLerp.ToString();

this.transform.GetChild(3).GetComponent<Text>().text = NormalStrength.ToString();

this.transform.GetChild(5).GetComponent<Text>().text = Refraction.ToString();

this.transform.GetChild(7).GetComponent<Text>().text = Lerp.ToString();

}

}

# Додаток ?. Лістинг програми до практичної роботи №?

### Код файлу (???)

### Код файлу (???)